Claims

We claim:

1.) A route determination algorithm with the properties that it routes on multiple levels concurrently using varying related route requirements, paths from lower levels act as simple links at higher levels, and returns a selection of routes of varying characteristics that all meet the route requirements for final route commitment.

2.) The method of Claim 1, wherein each level has the following level specific data sets:

a. A set of nodes

b. A set of links that interconnect the nodes

c. An array describing the interconnection of links and nodes

d. A level is either full duplex or half duplex

e. A set of metric properties used at this level

f. A cache fill algorithm used to fill the array of cache entries for this level

g. A resource reservation, selection, and commitment method

h. A cache of previously discovered approximate paths that have not yet been aged out.

3.) The method of Claim 2a, wherein each node/link has a set of primitive metric values that are used to calculate a path's aggregate values (by its cache fill algorithm (Claim 2.f)). The members of the set are different for each level (Claim 30). When a common cache fill algorithm is used (Claim 59), the step_over method will perform the aggregation (Claim 32).

4.) The method of Claim 2a, wherein each node/link has a set of resources that are used to interconnect the links to provide a data transmission path of multiple hops (i.e. using multiple cascaded fibers or wavelengths).

5.) The method of claim 2a, wherein the resources a node has determines whether it has a basic property that is used to determine the node's participation at a level. Example, if the level is Lambda switching level, then the node must have Lambda ports (and Lambda switching matrix by implication) to participate.

6.) The method of claim 5, wherein a node has a set of ports that serve as input or output for a data stream, the port has a line speed that is an operational characteristic, and it has the ability to connect any of its input ports to any output port to extend a data transmission path.

7.) The method of Claim 2a, wherein a node may have properties that allow it to participate at multiple levels and its participation at one level is not conditioned on its participation at any other level.

8.) The method of Claim 2b, wherein at the lowest operational level of the algorithm a link is either:

a.A set of single fibers that all have the same node terminations and can transmit data in the same direction.

b.A set of pairs of fibers that all have the same node terminations and can transmit data in both directions using one fiber in one direction and the other fiber in the reverse direction

9) The method of Claim 2b, wherein each link has a set of resources that are used to carry multiple data streams from its transmission ports to its receiving ports.

10.) The method of Claim 2b, wherein at levels above the lowest level, a link is a cache entry at the next lower level; possibly cascading to the lowest level.

11.) The method of Claim 2b, wherein each link has a set of primitive metric values that are used to calculate a path's aggregate values by the cache fill algorithm. The makeup of the set is defined by the level (Claim 30). When a common cache fill algorithm is used (Claim 59), the step_over method will perform the aggregation (Claim 32).

12) The method of Claim 11, wherein when a link is actually a set of cache entries from a lower level, it inherits the cache entries path aggregate metrics (with level dependent translation) as its link metrics.

13) The method of Claim 2b, wherein during cache fill (Claim 2f), a link has a set of pheromone aggregate metrics that are initialized at worst case values so that the first path probe of a non-transited link will always pass the improves condition (Claim 36), update

the value (Claim 36) and succeeding possible transits will only work if they can improve the value further.

14) The method of Claim 2b, wherein for half duplex links, all transmission ports are components of the same node and all receive ports are components of the same node.

15) The method of Claim 2c, wherein the neighbor array has an entry for each node at this level, each cell contains a list of pairs.

16) The method of Claim 15, wherein each pair is the pointer to a link and a pointer to a node (that is the other end of the link (i.e. A neighbor node)).

17) The method of Claim 15, wherein for full duplex links, the neighbor node will have a companion entry including the same link and itself.

18) The method of Claim 15, wherein this structure allows as many links between the same pair of nodes as there are unique links with differing routing characteristics (including DWDM transmission characteristics).

19) The method of Claim 15, wherein for half duplex nodes, the direction of travel is embedded in the neighbor structure. If the travel is from node A to node B using link 6, Then in the neighbor array, node A will have an entry [6,B] but node B will not have a

corresponding entry [6,A]. If there is a path from Node B to Node A it will use a different link whose direction is B to A.

20) The method of Claim 2d, wherein for each level, two independent caches are maintained; one for full duplex path requests, the other for half duplex path requests

21) The method of Claim 2h, wherein each cache may have sub-caches for route differentiation to preserve routing properties. For example Source A may have multiple links fanning out from it to neighbor nodes, each has a different set of DWDM lambda characteristics. We explore each of these in a single cache fill pass but we keep the results in different cache entries keyed off the DWDM lambda configuration characteristics (i.e. 128 channels by 1 Gigabit)

22) The method of Claim 2h, wherein the cache structure is a directory of directories. The upper directory key is (Source node index, sub-cache key) and the lower directory key is the destination node identifier

.

23) The method of Claim 22, wherein each destination level cache entry also contains a age property of when the cache was filled.

24) The method of Claim 23, wherein the age property is the current path index value (Claim 70) at the global level of the algorithm of the route request that causes the cache to be filled.

25) The method of Claim 24, wherein a cache entry's age is the difference between the current path index and the one recorded in Claim 23.

26) The method of Claim 2g, wherein each cache level has a get_path method that returns a set of possible paths from the source to the destination that meets the routing requirements.

27) The method of Claim 26, wherein each level's get_path method checks to see if the cache is empty or aged out and if it has, it attempts to get a cache from its next lower level and only if that fails will it fill its cache from that lower level and build an entry for the destination node and then attempt to fill the path request from that cache's entries.

28) The method of Claim 27, wherein each level's get_path method passes its results to the global get_path function which converts the relative path at that level to the fully filled in final path by using the top level's commit method (Claim 29).

29) The method of Claim 2g, wherein each cache level has a commit method that reserves resources for its level components (links or nodes) and then commits the lower level portions of its path. If that is successful it then commits the reserved resources and returns the final path fragment to its caller. If the lower levels are unsuccessful for any reason it returns a failure to the top level that can then choose a different path from the set it was given.

30) The method of Claim 2e, wherein a metric set for a level is made up of a set of primitive metrics that have a set of methods that are common to all. Each metric set shares the same set of methods. Application of one of these methods for the set generally applies the method to all the primitives that are within the set (the improves method is an exception). The set of primitives used at any level is independent of those used at other levels. A primitive metric may or may not be used in multiple levels.

31) The method of Claim 30, wherein the common methods are step_over, meets_requirements, improves, update, is_equal, better, copy, get_value.

32) The method of Claim 31, wherein "step_over" accepts a path aggregate metric (or metric set) and a node or link specific metric (or metric set) and produces a new path aggregate metric set that represents the path values when the path is extended over the node or link being considered.

33) The method of Claim 31, wherein "meets_requirements" accepts a path aggregate metric (or metric set) and a route requirements aggregate metric (or metric set) and returns a Boolean value whether the path does or does not meet the route requirements.

34) The method of Claim 31, wherein "improves" accepts a path aggregate metric (or metric set) and a pheromone aggregate metric (or metric set) and returns a Boolean value whether the path can improve on one or more critical primitive metrics that apply to this level of the algorithm.

35) The method of Claim 31, wherein "update" accepts a path aggregate metric (or metric set) that will improve upon a pheromone aggregate metric and the pheromone aggregate metric (or metric set) and records the improved values in the pheromone aggregate primitive metrics that will be improved.

36) The method of Claim 30, wherein general primitive metrics are one of the following types:

a) Boolean

b) minimums

c) additive

d) probability

e) sets of minimums

37) The method of Claim 36, wherein a subset of the primitives of a set participate in the "improves" method of the set.

38) The method of Claim 36a, wherein "step_over" a Boolean produces an AND of the two input values

39) The method of Claim 36a, wherein "meets_requirements" of a Boolean is the AND of the two input values

40) The method of Claim 36a, wherein "improves" of a Boolean is true if the path aggregate is true and the pheromone value is False

41) The method of Claim 36a, wherein "update" of a Boolean substitutes True for a pheromone's value if the path aggregate is True and the pheromone value is False

42) The method of Claim 36b, wherein "step_over" a minimum produces a minimum of the two input values

43) The method of Claim 36b, wherein "meets_requirements" of a minimum returns True if the path aggregate is larger than or equal to the requirements aggregate.

44) The method of Claim 36b, wherein "improves" of a minimum returns True if the path aggregate is larger than the pheromone aggregate and False otherwise.

45) The method of Claim 36b, wherein "update" of a minimum replaces the pheromone value with the path value IF the path value improves the pheromone value

46) The method of Claim 36c, wherein "step_over" an additive produces a sum of the two input values

47) The method of Claim 36c, wherein "meets_requirements" of an additive returns True if the requirement aggregate is equal to or larger than the path aggregate.

48) The method of Claim 36c, wherein "improves" of an additive returns True if the path aggregate is less than the pheromone aggregate and False otherwise.

49) The method of Claim 36c, wherein "update" of an additive replaces the pheromone value with the path value IF the path value improves the pheromone value

50) The method of Claim 36d, wherein "step_over" a probability produces a multiplication of the two input values

51) The method of Claim 36d, wherein "meets_requirements" of a probability returns True if the requirement aggregate is less than or equal the path aggregate.

52) The method of Claim 36d, wherein "improves" of a probability returns True if the path aggregate is larger than the pheromone aggregate and False otherwise.

53) The method of Claim 36d, wherein "update" of a probability replaces the pheromone value with the path value IF the path value improves the pheromone value

54) The method of Claim 36e, wherein "step_over" a set of minimums produces a set of minimums of which each indices value of the output is the minimum of the corresponding indices of the input values. To be clear, if a[3] is 4 and b[3] is 6, then result[3] will be 4.

55) The method of Claim 36e, wherein "meets_requirements" of a set of minimums returns True if any indices value of the path aggregate is greater than or equal to the requirements aggregates corresponding indices value.

56) The method of Claim 36e, wherein "improves" of a set of minimums returns True if any pheromone aggregate's indices value is zero and its corresponding path aggregate's indices value is greater than zero.

57) The method of Claim 36e, wherein "update" of a set of minimums replaces the pheromone indices value with the corresponding path indices value if the path value is greater than the pheromone value.

58) The method of Claim 2f, wherein each level may have a custom cache fill algorithm or it may use a common one that depends upon the level specific metrics to customize its behavior.

59) The method of Claim 58, wherein the common cache fill algorithm uses the "step_over","meets_requirements", "improves", and "update" methods (Claim 31) to guide the level specific cache filling.

60) The method of Claim 2f, wherein the cache fill algorithm starts from the source and explores all one hop paths to neighbors, followed by 2 hop paths, then 3 hop paths, ….., until there are no more paths to explore. This is the classical maze runner algorithm that can be found in many computer science text books.

61) The method of Claim 60, wherein each iteration of the algorithm works off of a ring structure developed as an output by the preceding iteration and produces a ring structure to be used as input for the next iteration.

62) The method of Claim 61, wherein the ring is a directory of value and path combinations for the next intermediate node in an outward probe. The key to the directory is the index of the node at this level.

63) The method of Claim 62, wherein each intermediate node iteratively examines its neighbors for possible extensions to its set of partial paths

64) The method of Claim 63, wherein each time the algorithm "steps over" a link and next node successfully, it writes all the paths that it is carrying with it to the cache.

65) The method of Claim 64, wherein a path does not output to the output ring when:

a) The intermediate node or link is disabled

b) The link lambda parameters do not match the paths lambda parameter

c) The path already contains the link or node

d) The intermediate path exceeds any route requirement metric

e) The intermediate path encounter a link that has already been transited and it can't improve upon the link's pheromone value

66) The method of Claim 65e, wherein each successive transit of a link must improve upon one of several primitive pheromone values left by previous transits. These are defined in Claim .2d.

67) The method of Claim 2g, wherein each level of the commits method (reserve, select, and commit method) cascades down to commit lower level components before it returns a committed path to its upper levels.

68) The method of Claim 67, wherein when higher level paths find that a full path that it is trying to commit cant be completed, it has to release the committed or reserved lower level segments of the path before examining the next possibility (Claim 29).

69) The method of Claim 1, wherein each level is associated with specific routing properties and all paths in the level's cache (Claim 2d) will possess that routing property and all links and nodes participating in the level will preserve the property. For example, at the Lambda routing level, a path from end to end will use the same DWDM wavelength and be bounded by the specific portion of the light spectrum that is that wavelength. All nodes and links along the path will preserve this property.

70) The method of Claim 1, wherein the algorithm maintains a global index that is incremented for each path request at any level. This is used to record when cache entries are filled and to age them out.

71) The method of Claim 1, wherein the algorithm has a "find_route" method that accepts a number of parameters such as source, destination, routing requirements, and initial aggregate values and returns either an acceptable route or a failure indicator.

72) The method of Claim 71, wherein the initial aggregate values are provided so that routes from multiple administrative domains may be stitched together into a single path that intermediate Path Determination Algorithms can determine the acceptability of paths they are considering.

73) The method of Claim 71, wherein the routing parameters instantiation sets the find_route method to call the highest level of its cache's get_path method and then calls its select_path method to commit a path from the current cache entries.

74) The method of Claim 71, wherein the algorithm maintains a set of lambda configurations that includes all configurations seen at any link in its data sets for the Lambda and data regeneration levels. This is used to help it decide lambda configuration suitability for paths.

75) Claim 1 is not in any way sensitive to any data framing format for the data streams that traverse its routes, nor is it dependent upon any specific formats for exchange of data about the information in its data structures, nor is it specific to any methods of commanding switching and transmission gear to create route segments. The invention is

opaque to the contents or the framing/modulation structure of the data that flows along the path it establishes.

76) An implementation of Claim 1 that is specific to the problem of routing in an optical network. It introduces an assignment of levels to specific optical routing properties, node and link resources that have these properties, metrics that allow route requirements based upon these properties, and path selection algorithms that optimize the use of network resources to deliver these properties. This is accomplished using the general cache_level_base and the level specific cache_level instances.

77) The method of Claim 76, wherein the levels are from top to bottom:

a) Multiplexor level that supports add and dropping sub-channels of varying sizes at particular switch ports in the network.

b) O-E-O switching level that supports switching of optical routes using bandwidth limited electronic switching technologies such as TDM

c) lambda translation level that supports discrete or built in transitions from one lambda to another through non TDM techniques such as VCSEL units

d) lambda signal regeneration that does not require lambda translation or TDM switching

e) Lambda level routing of lambdas

f.) Whole Fiber level routing including all lambda within a fiber

78) The method of Claim 76, wherein the cache for the lambda signal regeneration and Lambda levels use sub-caching (Claim 26) for paths of differing lambda configurations. For example, one sub-cache may contain paths for 128 channels at 2.5 gigabits per channel while another may have paths for 96 channels at 10 gigabits per channel.

79) The method of Claim 78, wherein the primitive metrics are:

a) Route Length (additive type-Claim 36c) is the length of a fiber or path in meters.

b) Insertion loss (additive type-Claim 36c) is the decibel of signal loss of a fiber, switch traversal or a path

c) Regenerator availability (minimum type-Claim 36b) is the number of signal regenerators available at a node along a path that has the least of all nodes in the path; for a node it is the number of signal generators that the node has available for assignment.

d) Lambda translator availability(minimum type-Claim 36b) is the number of lambda translators available at a node along a path that has the least of all nodes in the path; for a node it is the number of lambda translators that it has available for assignment.

e) Lambda assignment availability (set of minimums-Claim 36e) is the minimum number of assignments available for each lambda at any link along a path.

f) Latency (additive –Claim 36c) is the estimated delay a signal will experience transiting a path in milliseconds.

g) Jitter (additive-Claim 36c) is the estimated variance in delay a signal will experience transiting a path in milliseconds.

h) Bandwidth availability (minimum-Claim 36b) is the minimum bandwidth availability at any link or node that a path transits

i) Bit Error rate probability (probability-Claim 36d) is the probability that a signal that transits the path will see no bit errors in any standard time period.

80) The method of Claim 76, wherein at the lambda switching level a link is a bundle of single fibers or pairs of fibers, each with its own insertion loss that may vary even when all fibers in the bundle are the same length. We estimate the nominal insertion loss as the mean between the fibers with remaining lambda assignments that are the maximum and minimum for the bundle (specialization of Claim 12).

81) The method of Claim 76, wherein Lambda level metric set (Specialization of Claim 30) has insertion loss, lambda assignment availability, lambda translator availability, regenerator availability, route length, available bandwidth, latency, jitter, and bit error rate probability. Improves only looks at Insertion loss, lambda assignment availability, and route length.

82) The method of Claim 76, wherein Regenerator level metric set (Specialization of Claim 30) has lambda assignment availability, lambda translator availability, regenerator availability, route length, available bandwidth, latency, jitter, and bit error rate probability. Improves only looks at lambda assignment availability, regenerator availability, and route length.

83) The method of Claim 76, wherein Lambda Translator level metric set (Specialization of Claim 30) has insertion loss, lambda assignment availability, lambda translator availability, regenerator availability, route length, available bandwidth, latency, jitter, and bit error rate probability. Improves only looks at lambda translator availability, regenerator availability, and route length.

84) The method of Claim 76, wherein O-E-O level metric set (Specialization of Claim 30) has lambda translator availability, regenerator availability, route length, available bandwidth, latency, jitter, and bit error rate probability. Improves only looks at Lambda translator availability, regenerator availability, available bandwidth, and route length.

85) The method of Claim 76, wherein Multiplexor level metric set (Specialization of Claim 30) has lambda translator availability, regenerator availability, route length, available bandwidth, latency, jitter, and bit error rate probability. Improves only looks at Lambda translator availability, regenerator availability, available bandwidth, and route length.

86) The method of Claim 76, wherein cache age out for Lambda level is when cache age (Specialization of Claim 25) is greater than the total of minimum lambda assignments available for the minimum path when recorded.

87) The method of Claim 76, wherein cache age out for regenerator level is when cache age (Specialization of Claim 25) is greater than the minimum regenerators available for the minimum path when recorded.

88) The method of Claim 76, wherein cache age out for lambda translator level is when cache age (Specialization of Claim 25) is greater than the minimum lambda translators available for the minimum path when recorded.

89) The method of Claim 76, wherein cache age out for O-E-O level is when cache age (Specialization of Claim 25) is greater than the minimum available estimated connections (a function of minimum available bandwidth) for the minimum path when recorded.

90) The method of Claim 76, wherein cache age out for multiplexor level is when cache age (Specialization of Claim 25) is greater than the minimum available sub channels (a function of minimum available bandwidth) for the minimum path when recorded.

91) The method of Claim 76, wherein regenerator level path selection is a 4 step selection of (Specialization of 2g):

a) order paths on fewest required generators, best lambda configuration, best lambda availability

b) pick and reserve generators at nodes

c) pick and reserve lambda assignment

d) iterate over Lambda level fiber selection with lambda assignment as input until all links have fiber assignments.

e) commit fiber assignments, lambda assignment, lambda configuration, and regenerators

92) The method of Claim 91d, wherein Lambda level fiber selection is an 5 step selection of (Specialization of 2g):

a) For the chosen lambda calculate the minimum possible end to end insertion loss by aggregating the minimum nominal insertion loss from the fibers in each link which have that lambda available.

b) calculate a target insertion loss of min_loss + (1/2 * (max_loss – min_loss)) where max_loss is the insertion loss from the routing requirement metric set for the path selection.

c) assign a random order to the links of the path

d) iterate through all combinations of fibers using the random ordering; recording the closest to the target until a match to the target is found

e) if no match is found return the closest fit

93) The method of Claim 91a, wherein Lambda configuration selection is a multi-step task:

a) order the configurations in descending order of transmission speed followed by channel count

b) iterate from the bottom; ignoring all transmission speeds less than routing requirements

c) exhaust all possibilities of a path at a configuration before moving to the next

d) if transmission speed is greater than the routing requirements check if the local policy table allows this combination

94) The method of Claim 91c, wherein Lambda assignment selection is a multi-step task:

a) precedence order the paths; largest number of non-zero lambda assignments, deepest single lambda assignment, largest count with the maximum depth, lowest lambda index with the maximum depth.

95) The method of Claim 76, wherein lambda translator level path selection is a multi step task (Specialization of 2g):

a) order paths on fewest required translators, fewest required generators, best lambda configuration, best lambda availability, shortest route length

b) pick and reserve lambda translators at each node

c) find and commit generator level links

d) commit lambda translators

96) The method of Claim 76, wherein O-E-O level path selection is a multi step task (Specialization of 2g):

a) order paths on fewest required translators, fewest required generators, largest available bandwidth, shortest route length, best bit error rate possibility

b) reserve the required bandwidth at each node

d) commit lambda translator level links

e) commit the bandwidth for the nodes

97) The method of Claim 76, wherein Multiplexor level path selection is a multi step task (Specialization of 2g):

a) order paths on fewest required translators, fewest required generators, largest available bandwidth, shortest route length, best bit error rate possibility

b) reserve the required bandwidth at each node

d) commit O-E-O level links

e) commit the bandwidth at the nodes

98) The method of Claim 76, wherein the entries in each levels neighbor array for the Lambda level and the regenerator level allows as many links between a pair of nodes as there are unique lambda configurations installed on fibers that connect the pair (specialization of Claim 67).

99) The method of Claim 76, wherein ports terminates level specific properties (specialization of Claim 6). Whole fiber ports source/terminate all the lambdas of a fiber collectively at the fiber switching level of the algorithm using the fiber-switching matrix. OADM ports source/terminate a subset of the lambdas of a fiber collectively at the fiber switching level of the algorithm. Lambda ports source and terminate individual lambdas and maintain lambda data integrity along the path using the OEO and/or lambda switching matrix plus regenerators and/or lambda translators. Multiplexor Ports match sub-lambda standard optical multiplexor channels to a lambda using the OEO switching matrix.

100) The method of Claim 76, wherein as links inherit characteristics from the cache entries at the next lower level, the metrics of the lower levels are translated into metrics at the higher levels (specialization of Claim 12).

101) The method of Claim 100, wherein For the generator level we drop use of the insertion loss primitive metric. All other values translate as is. Routing requirements metric set will trigger termination of path search when the aggregated insertion loss exceeds its value (the maximum allowable insertion loss). This will preserve that no path will allow signal level to drop below the level where it is so degraded as to be unusable.

102) The method of Claim 76, wherein Nodes that participate in LAMBDA routing level have switching matrices that maintain a lambdas spectrum properties as they transit the matrix such that a receiver terminating a multi-hop path will not be able to determine whether the transmission was from a direct neighbor or not. The only allowable deviation from this condition is that the signal strength may be degraded more than it would if it were traversing a fiber of length greater than the path across the switching matrix, but the degradation may not be sufficient to significantly impede the receivers ability to discern the data transmitted over the path (Specialization of claim 3).

103) The method of Claim 102, wherein Nodes that participate in the regeneration level will also maintain a lambdas spectrum properties. The nodes have a property that they will provide signal regeneration to compensate for any aggregated signal distortion that

may have occurred in the path prior to the node. Thus paths at this level or above do not need to be concerned with insertion loss, since the level below will insure insertion of regenerators when appropriate.

104) The method of Claim 103, wherein Nodes that participate in the lambda translation level will be able to match output ports that use one lambda with input ports that use a different lambda while preserving the data integrity of the data stream using the lambdas as transmission media. There are 2 mechanisms to insure this. The first is the lambda_assignment_availability metric. When there is no available lambda assignments, a new segment will be terminated with a lambda translator. The second is a test in the cache fill algorithm that insures that a path will only extend along links that have the same lambda configuration as that of the source link.

105) The method of Claim 104, wherein Nodes that participate in the O-E-O level will preserve the data integrity of the data streams that traverse them.

106) The method of Claim 105, wherein Nodes that participate in the Multiplexor level will provide add/drop multiplexing capability while preserving the integrity of the aggregate data streams that transit the node.

107) The method of Claim 31, wherein is_equal accepts two metrics and produces a boolean '1' if they are or a boolean '0' if not.

108) The method of Claim 31, wherein better accepts a path aggregate metric (or metric set) and a second metric (or metric set) and produces a boolean '1' if the second is better than the first.

109) The method of Claim 31, wherein copy accepts an aggregate metric (or metric set) and creates a new metric that is a copy of the first.

110.) The method of Claim 31, wherein get_value of a primitive metric return its current value.

111.) The method of Claim 36, wherein a subset of the primitives of a set participate in the meets_requirement method of the set.

112) The method of Claim 76, wherein Whole fiber level metric set (Specialization of Claim 30) has insertion_loss, route_length, available bandwidth, lambda translator availability, regenerator availability, route length, latency, jitter, and bit error rate probability. Improves only looks at insertion_loss and route length.

113) The method of Claim 76, wherein cache age out for whole fiber level is when cache age (Specialization of Claim 25) is greater than the maximum of the minimum fiber count for each path in the cache.

Inventors: Matthews, Wallace E. (Mendon, Mass)

Assignee: Matthews, Wallace E. (Mendon, Mass)

Appl No.:

Filed:

References Cited

U.S. Patent Documents

4201889	06-May-1980	Lawrence et. al.
4565903	21-Jan-1982	Riley
4873517	10-Dec-1989	Baratz et. al.
4979118	18-Dec-1990	Kheradpir
4987536	22-Jan-1991	Humblet
4991204	05-Feb-1991	Yamamoto
5218602	08-Jun-1993	Grant
5233607	03-Aug-1993	Barwig
5317562	31-May-1994	Nardin
5321815	14-Jun-1994	Bartalanzo et.al.
5452295	19-Sep-1995	Natarajan
5463620	31-Oct-1995	Sriram
5519836	21-May-1995	Gawlick et.al.

5521910	28-May-1996	Matthews
5526414	11-Jun-1996	Bedard et.al.
5539815	23-Jul-1996	Samba
5561790	01-Oct-1996	Fusaro
5600638	04-Feb-1997	Bertin et. al.
5600794	04-Feb-1997	Callon
5751706	12-May-1998	Land et. al.
5832069	03-Nov-1998	Waters et. al.
5805593	08-Sep-1998	Busche
5893081	06-Apr-1999	Poppen
5937397	10-Aug-1999	Callaghan
5999517	07-Dec-1999	Konig et. al.
6016306	18-Jan-2000	Le Boudec et. al.
6016485	18-Jan-2000	Amakawa
6084858	04-Jul-2000	Matthews et. al.
6151327	21-Nov-2000	Sofman et. al.
6205154	20-Mar-2001	Schmidt et. al.
6205484	20-Mar-2001	Eriksson

Foreign Patent Documents

Other References

Coral Broadband Enterprise Switch, Product Literature, Coral Network Corp. Marlborough, MA (1994)

C. J. Wang, E. P. K. Tsang, "Solving Constraint Satisfaction Problems Using Neural Networks", Second International Conference on Artificial Neural Networks, Nov. 18-20, 1991.

S. Selman, H. Levesque, D. Mitchell, "A New Method for Solving Hard Satisfiability Problems", AAAI-92, Proceedings Tenth National Conference on Artificial Intelligence, Jul. 12-16, 1992.

Maruyama et al, "Solving Combinatorial Constraint Satisfaction and Optimization Problems using Sufficient Conditions for Constraint Violation", Proc Int'l Symposium on AI, Nov. 13-15 1991, pp. 269-275.

Smith et al, "Combining Constraint Satisfaction and Local Improvement Algorithm to Construct Anaesthetists Rates" Proc 8th Conf on AI for Applications, Mar. 2-6, 1992, pp.106-112.

Wang et al, "Solving Constraint Satisfaction Problems Using Neural Networks" and Int'l Conf on Neural Networks, Nov. 18-20, 1991.

Miltal et al, "Dynamic Constraint Satisfaction Problems", Proc 8th Nat'l Conference on AI, vol. 1 pp. 25-32, Jul. 29-Aug. 3 1990.

This Patent was developed independently of any Federally sponsored research. It is solely the work of Wallace Matthews of 41 Kinsley Lane, Mendon, Mass. Currently self employed and an independent inventor.

An example software listing is being supplied on CD Roms. The CDRom is CD-R machine Format produced on an IBM Compatible PC running Windows-2000 using the MicroSoft Windows NT file System formats. All files are Ascii Text files and viewable via any Unix or Windows compatible Text Editor. The files are Python Source code files with a ".py" extension. Python is the chosen language because it is very "readable" yet compact. For those not familiar with Python, it has many of the characteristics of Java, yet is easier for a novice programmer to learn and to understand.

File Names	File Size in Bytes	Original Creation Date
Route_1.py	15,000	6/17/2001 8:08PM
Route_common.py	3,000	6/17/2001 7:56PM
Policy.py	1,000	5/20/2001 9:59PM
Path.py	7,000	6/17/2001 2:37PM
Node.py	23,000	6/17/2001 7:36PM

Metric.py	16,000	6/2/2001 8:44PM
Link.py	24,000	6/17/2001 8:42PM
Cache.py	50,000	6/17/2001 8:29PM